

getdata.pl Documentation

Submitted under Task Agreement GSFC-CT-2

Cooperative Agreement Notice (CAN) CAN-00OES-01

Increasing Interoperability and Performance of
Grand Challenge Applications in the Earth, Space, Life, and Microgravity Sciences

May 7, 2004

Revision 1.0

History:

| Revision | Summary of Changes | Date |
|----------|--------------------------|-------------|
| 1.0 | Milestone "G" submission | May 7, 2004 |



National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Contents

| | |
|---|----------|
| 1 Routine/Function Prologues | 3 |
| 1.0.1 getdata.pl (Source File: <i>getdata.pl</i>) | 3 |
| 1.0.2 ProcessArgs (Source File: <i>getdata.pl</i>) | 4 |
| 1.0.3 SetEnv (Source File: <i>getdata.pl</i>) | 5 |
| 1.0.4 BuildNames (Source File: <i>getdata.pl</i>) | 6 |
| 1.0.5 CreateDir (Source File: <i>getdata.pl</i>) | 8 |
| 1.0.6 BuildHoF (Source File: <i>getdata.pl</i>) | 8 |
| 1.0.7 getsand (Source File: <i>getdata.pl</i>) | 9 |
| 1.0.8 getclay (Source File: <i>getdata.pl</i>) | 10 |
| 1.0.9 getalbedo (Source File: <i>getdata.pl</i>) | 10 |
| 1.0.10 getcolor (Source File: <i>getdata.pl</i>) | 11 |
| 1.0.11 getgfrac (Source File: <i>getdata.pl</i>) | 12 |
| 1.0.12 getmask (Source File: <i>getdata.pl</i>) | 12 |
| 1.0.13 getmaxsnalb (Source File: <i>getdata.pl</i>) | 13 |
| 1.0.14 gettbot (Source File: <i>getdata.pl</i>) | 14 |
| 1.0.15 getveg (Source File: <i>getdata.pl</i>) | 14 |
| 1.0.16 getgeos (Source File: <i>getdata.pl</i>) | 15 |
| 1.0.17 getgdas (Source File: <i>getdata.pl</i>) | 16 |
| 1.0.18 getmask1km (Source File: <i>getdata.pl</i>) | 17 |
| 1.0.19 getveg1km (Source File: <i>getdata.pl</i>) | 18 |
| 1.0.20 getelev (Source File: <i>getdata.pl</i>) | 18 |
| 1.0.21 getagrmel_sw (Source File: <i>getdata.pl</i>) | 19 |
| 1.0.22 getagrmel_lw (Source File: <i>getdata.pl</i>) | 20 |

1 Routine/Function Prologues

This perl script is used to retrieve data files from LIS' GrADS-DODS data server (GDS). It is invoked by the LIS executable via a call to the system routine; e.g.:

```
call system("opendap_scripts/getmask.pl "//ciam//" //&
           trim(lis%p%mfile)//" //&
           cslat//" //cnlat//" ///cwlon//" //celon)
```

Note: This script is not called directly, but rather through symbolic links. I.e.; getmask.pl points to getdata.pl. The getdata.pl script determines how it is called (in this example, as getmask.pl) to determine which data-set it must retrieve and how to parse the input arguments.

1.0.1 getdata.pl (Source File: getdata.pl)

USES:

```
use warnings;
use strict;
```

INPUT PARAMETERS:

```
my $iam;          # -- id of calling process
my $fullfilename; # -- name of file to retrieve (including path)
my $slat;         # -- southern latitude boundary of subset to retrieve
my $nlat;         # -- norhtern latitude boundary of subset to retrieve
my $wlon;         # -- western longitude boundary of subset to retrieve
my $elon;         # -- eastern longitude boundary of subset to retrieve
my $index;        # -- time-offset index (only for getgeos and agrmet)
```

OUTPUT PARAMETERS:

Writes subset of data into fullfilename

LOCAL VARIABLES:

```
my $filename;
my $filepath;
my $ctl;
my $ctlpath;

my $home_dir;
my $server;
my $log_dir;
my $GrADS;

my $data;
my %get;

my $m_or_q;      # -- month or quarter-season to retrieve (only for getgfrac and getalbedo)
```

CONTENTS:

```

&ProcessArgs(@ARGV);
&SetEnv;
&BuildNames;
&CreateDir;
&BuildHoF;

# Spread out gds requests
#`sleep $iam`;

# Remove the cache!
print " MSG: $0 -- Removing cache ( $iam )\n";
`rm -rf $home_dir/.dods_cache`;

# Make GDS request
print " MSG: $0 -- Making GDS request to server $server ( $iam )\n";
&{ $get{$data} };

print " MSG: $0 -- script done ( $iam )\n";

```

1.0.2 ProcessArgs (Source File: getdata.pl)

This subroutine processes the command line arguments to the main routine.

CONTENTS:

```

sub ProcessArgs
{
    my $called_as;
    my %data;

    %data = (
        "getsand"      => "SAND",
        "getclay"      => "CLAY",
        "getcolor"     => "COLOR",
        "gettbot"      => "TBOT",
        "getmask"       => "UMD Mask",
        "getveg"        => "UMD Veg",
        "getmask1km"   => "UMD Mask 1KM",
        "getveg1km"    => "UMD Veg 1KM",
        "getmaxsnalb"  => "MAXSNALB",
        "getalbedo"    => "ALBEDO",
        "getgfrac"     => "GFRAC",
    );
}

```

```

    "getgeos"      => "GEOS",
    "getgdas"      => "GDAS",
    "getelev"       => "ELEV",
    "getagrmetsw"  => "AGRMET_SW",
    "getagrmetsw"  => "AGRMET_LW",
);

( $called_as = $0 ) =~ s/.*\.(.*).pl:$1: ;
$data = $data{$called_as} or die "ERR: $0 -- Called incorrectly";

if ( $data eq "GEOS" )
{
    ($iam, $fullfilename, $index, $slat, $nlat) = @_;
}
elsif ( $data eq "AGRMET_SW" or $data eq "AGRMET_LW" )
{
    ($iam, $fullfilename, $index, $slat, $nlat, $wlon, $elon) = @_;
}
else
{
    ($iam, $fullfilename, $slat, $nlat, $wlon, $elon) = @_;
}
}

```

1.0.3 SetEnv (Source File: getdata.pl)

This subroutine determines which system this script is being run on and sets several global variables.

CONTENTS:

```

sub SetEnv
{
    my $system;

    chomp( $system = `uname -s` );

    # If $system is "Linux" then this script assumes that it is running on
    # LIS' Linux cluster.  If you are running on your own Linux system
    # then see the corresponding else-block to determine which variables to set.

    if ( $system eq "Linux" ) # running on LIS' linux cluster
    {

        print " MSG: $0 -- Determining GDS ( $iam )\n";
    }
}

```

```

do {
    open SERVER, "</data1/GDS/list.txt"; # this file contains a list of
    chomp( $server = <SERVER> );           # available servers, grab the
    close SERVER;                          # first one in the list
    undef $server unless $server =~ m/\S/;
    sleep 1;
} until defined $server;
##$server = "x7";
print " MSG: $0 -- Using GDS $server ( $iam )\n";

$home_dir = "./";
open(CARD_FILE, "<lis.crd") or die "ERR: Cannot open lis.crd\n";
while ( <CARD_FILE> ) {
    next unless m/opendap_data_prefix/ ;
    chomp;
    ( my $dummy, $home_dir ) = split(/=/);
}
close(CARD_FILE);
$home_dir =~ s/.*/(.*)\.*$/1;
$log_dir = "$home_dir/$iam";

$GrADS="/data1/GRADS-packages/grads-1.9b2-src-gsfc/bin/gradsdods";
}
else
{
    $server="lisdata.gsfc.nasa.gov";
    $home_dir ".";
    $log_dir ".";
    $GrADS="/u/jvg/local/grads-1.8sl11/bin/grads";
}
}
}

```

1.0.4 BuildNames (Source File: getdata.pl)

This subroutine takes the full file name of the data to be retrieved and breaks this name into the file path, file name, name of GrADS descriptor file, and path to the GrADS descriptor file.

CONTENTS:

```

sub BuildNames
{
    my $tmpctl;
    #filepath=${fullfilename%/*}

```

```

#filename=${fullfilename##*/}
#ctlpath=${filepath#/BCS/}
#ctl=${filename%.*}

if ( $data eq "SAND" or $data eq "CLAY" or $data eq "COLOR" or
     $data eq "MAXSNALB" or $data eq "TBOT" )
{
    ($filepath = $fullfilename) =~ s:(.*)/.*:$1: ;
    ($filename = $fullfilename) =~ s:.*/: ;
    ($ctlpath = $filepath)      =~ s:.*/BCS/: ;
    ($ctl      = $filename)    =~ s:(.*)\..*:.$1: ;
}
elsif ( $data eq "GEOS" or $data eq "AGRMET_LW" or $data eq "AGRMET_SW")
{
    ($filepath = $fullfilename) =~ s:(.*)/.*:$1: ;
    ($filename = $fullfilename) =~ s:.*/: ;
}
elsif ( $data eq "GDAS" )
{
    ($filepath = $fullfilename) =~ s:(.*)/.*:$1: ;
    ($filename = $fullfilename) =~ s:.*/: ;
    ($ctlpath = $filepath)      =~ s:.*/(.*):$1: ;
    ($ctl      = $filename)    =~ s:(.*)\..*:.$1: ;
}
elsif ( $data eq "UMD Mask" or $data eq "UMD Veg" or
        $data eq "UMD Mask 1KM" or $data eq "UMD Veg 1KM" or
        $data eq "ELEV" )
{
    ($filepath = $fullfilename) =~ s:(.*)/.*:$1: ;
    ($filename = $fullfilename) =~ s:.*/: ;
    ($ctlpath = $filepath)      =~ s:.*/GVEG/: ;
    ($ctl      = $filename)    =~ s:(.*)\..*:.$1: ;
}
elsif ( $data eq "ALBEDO" or $data eq "GFRAC" )
{
    ($filepath = $fullfilename) =~ s:(.*)/.*:$1: ;
    ($filename = $fullfilename) =~ s:.*/: ;
    ($ctlpath = $filepath)      =~ s:.*/BCS/: ;
    #($tmpctl = $filename)      =~ s:(.*)\..*:.$1: ;
    #($ctl    = $tmpctl)        =~ s:(.*)_.*_(.*):$1_$2: ;
    ($ctl      = $filename)    =~ s:(.*)_.*:$1: ;
    ($m_or_q   = $filename)    =~ s:.*_(.*)\..*:.$1: ;
}
print " DBG: $0 -- fullfilename $fullfilename ( $iam )\n";
print " DBG: $0 -- filepath $filepath ( $iam )\n";
print " DBG: $0 -- filename $filename ( $iam )\n";
unless ( $data eq "GEOS" or $data eq "AGRMET_LW" or $data eq "AGRMET_SW" )

```

```

{
    print " DBG: $0 -- ctlpath $ctlpath ( $iam )\n";
    print " DBG: $0 -- ctl $ctl ( $iam )\n";
}
}

```

1.0.5 CreateDir (Source File: *getdata.pl*)

This subroutine creates the directory the write the retrieved data into.

CONTENTS:

```

sub CreateDir
{
    print " MSG: $0 -- Fetching $data data ( $iam )\n";

    if ( -e $filepath )
    {
        print " MSG: $0 -- $data directory $filepath exists ( $iam )\n";
    }
    else
    {
        print " MSG: $0 -- Creating $data directory $filepath ( $iam )\n";
        unless ( system("mkdir -p $filepath") )
        {
            print " MSG: $0 -- Created directory $filepath ( $iam )\n";
        }
        else
        {
            print " MSG: $0 -- Error occured while creating directory $filepath ( $iam )\n";
            exit 1
        }
    }
}

```

1.0.6 BuildHoF (Source File: *getdata.pl*)

This subroutine creates a hash of functions to be used to call the appropriate data retrieval subroutine.

CONTENTS:

```

sub BuildHoF
{
    %get = (
        "SAND"      => \&getsand,
        "CLAY"      => \&getclay,
        "COLOR"     => \&getcolor,
        "ALBEDO"    => \&getalbedo,
        "GFRAC"     => \&getgfrac,
        "UMD Mask"   => \&getmask,
        "UMD Mask 1KM" => \&getmask1km,
        "MAXSNALB"  => \&getmaxsnalb,
        "TBOT"       => \&gettbot,
        "UMD Veg"    => \&getveg,
        "UMD Veg 1KM" => \&getveg1km,
        "GEOS"       => \&getgeos,
        "GDAS"       => \&getgdas,
        "ELEV"       => \&getelev,
        "AGRMET_SW"  => \&getagrmet_sw,
        "AGRMET_LW"  => \&getagrmet_lw,
    );
}

```

1.0.7 getsand (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the “sand fraction” soil data.
CONTENTS:

```

sub getsand
{
my $gradsstout;

$gradsstout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d sand
quit
EOB`;

```

```
open(LOG, ">>$log_dir/sand.$iam.log");
print LOG $gradsstdout;
close(LOG);
}
```

1.0.8 getclay (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the “clay fraction” soil data.
CONTENTS:

```
sub getclay
{
my $gradsstdout;

$gradsstdout =
`$GrADS -bl <<EOF
sdfopen http://$server:9095/dods/LIS_Params/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d clay
quit
EOF`;

open(LOG, ">> $log_dir/clay.$iam.log");
print LOG $gradsstdout;
close(LOG);
}
```

1.0.9 getalbedo (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the albedo data.
CONTENTS:

```
sub getalbedo
{
my $gradsstdout;
```

```
$gradsstdout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/${ctlpath}/${ctl}
set fwex
set t $m_or_q
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d alb
quit
EOB`;

open(LOG, ">> $log_dir/alb.$iam.log");
print LOG $gradsstdout;
close(LOG);
}
```

1.0.10 getcolor (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the soil color data.
CONTENTS:

```
sub getcolor
{
my $gradsstdout;

$gradsstdout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d soicol
quit
EOB`;

open(LOG, ">>$log_dir/color.$iam.log");
print LOG $gradsstdout;
close(LOG);
```

}

1.0.11 getgfrac (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the greenness fraction data.

CONTENTS:

```
sub getgfrac
{
my $gradsstout;

$gradsstout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/${ctlpath}/${ctl}
set fwex
set t $m_or_q
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d gfrac
quit
EOB`;
open(LOG, ">>$log_dir/gfrac.$iam.log");
print LOG $gradsstout;
close(LOG);
}
```

1.0.12 getmask (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the UMD land/sea mask data.

CONTENTS:

```
sub getmask
{
my $gradsstout;

$gradsstout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/GVEG/${ctlpath}/${ctl}
```

```

set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d tal
d nol
d mask
quit
EOB';
open(LOG, ">>$log_dir/mask.$iam.log");
print LOG $gradsstdout;
close(LOG);
}

```

1.0.13 getmaxsnalb (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the maximum snow albedo data.
CONTENTS:

```

sub getmaxsnalb
{
my $gradsstdout;

$gradsstdout =
`$GrADS -bl <<EOB
sdopen http://$server:9095/dods/LIS_Params/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d snalb
quit
EOB';
open(LOG, ">>$log_dir/snalb.$iam.log");
print LOG $gradsstdout;
close(LOG);
}

```

1.0.14 gettbot (Source File: *getdata.pl*)

This subroutine makes a system call to GrADS to retrieve the “bottom temperature” data.
 CONTENTS:

```
sub gettbot
{
my $gradsstout;

$gradsstout =
`$GrADS -bl <<EOF
sdfopen http://$server:9095/dods/LIS_Params/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d tbot
quit
EOF`;
open(LOG, ">>$log_dir/tbot.$iam.log");
print LOG $gradsstout;
close(LOG);
}
```

1.0.15 getveg (Source File: *getdata.pl*)

This subroutine makes a system call to GrADS to retrieve the UMD Vegetation classification data.

CONTENTS:

```
sub getveg
{
my $gradsstout;

$gradsstout =
`$GrADS -bl <<EOF
sdfopen http://$server:9095/dods/LIS_Params/GVEG/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
```

```

set gxout fwrite
d tal
d nol
d veg1
d veg2
d veg3
d veg4
d veg5
d veg6
d veg7
d veg8
d veg9
d veg10
d veg11
d veg12
d veg13
quit
EOB';
open(LOG, ">>$log_dir/veg.$iam.log");
print LOG $gradsstdout;
close(LOG);
}

```

1.0.16 getgeos (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the GEOS forcing data.
CONTENTS:

```

sub getgeos
{
my $gradsstdout;

$gradsstdout =
'$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Forcing/GEOS/BEST_LK/geos3new
set fwex
set t $index
set x 1 360
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d t2m
d q2m
d radswg

```

```

d lwgdown
d u10m
d v10m
d ps
d preacc
d precon
d albedo
d sfctyp
d snow
d gwet
d t10m
d q10m
quit
EOB';
open(LOG, ">>$log_dir/geos.$iam.log");
print LOG $gradsstdout;
close(LOG);
}

```

1.0.17 getgdas (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve GDAS forcing data.
CONTENTS:

```

sub getgdas
{
my $gradsstdout;

$gradsstdout =
'$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Forcing/GDAS/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d albdosfc
d cpratsfc
d dlwrfsfc
d dswrfsfc
d gfluxsfc
d lhtflsfc
d pevprsfc

```

```

d pratesfc
d pressfc
d shtflsfc
d soilwsoilt
d soilwsoilm
d spfh2m
d tmprfc
d tmp2m
d tmpsoilt
d tmpsoilm
d ugrd10m
d ulwrfsfc
d uswrfsfc
d vgrd10m
d watrsfc
d weasdsfc
quit
EOB';
open(LOG, ">>$log_dir/gdas.$iam.log");
print LOG $gradsstout;
close(LOG);
}

```

1.0.18 getmask1km (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the UMD 1km land/sea mask data.

CONTENTS:

```

sub getmask1km
{
my $gradsstout;

$gradsstout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/GVEG/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d mask
quit
EOB`;

```

```
EOB';
open(LOG, ">>$log_dir/mask.$iam.log");
print LOG $gradsstdout;
close(LOG);
}
```

1.0.19 getveg1km (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the UMD 1km Vegetation classification data.

CONTENTS:

```
sub getveg1km
{
my $gradsstdout;

$gradsstdout =
'`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/GVEG/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d veg
quit
EOB';
open(LOG, ">>$log_dir/veg.$iam.log");
print LOG $gradsstdout;
close(LOG);
}
```

1.0.20 getelev (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the elevation correction data.

CONTENTS:

```
sub getelev
{
```

```

my $gradsstdout;

$gradsstdout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Params/GVEG/${ctlpath}/${ctl}
set fwex
set t 1
set x $wlon $elon
set y $slat $nlat
set fwrite -be -sq $fullfilename
set gxout fwrite
d diff
quit
EOB`;
open(LOG, ">>$log_dir/elev.$iam.log");
print LOG $gradsstdout;
close(LOG);
}

```

1.0.21 getagrmet_sw (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the AGRMET shortwave radiation data.

CONTENTS:

```

sub getagrmet_sw
{
my $gradsstdout;

$gradsstdout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Forcing/AGRMET-SWDN/swdn
set fwex
set t $index
set x 1 1440
set y 1 600
set fwrite -be -sq $fullfilename
set gxout fwrite
d swdn
quit
EOB`;
open(LOG, ">>$log_dir/agrmet_sw.$iam.log");
print LOG $gradsstdout;
close(LOG);
}

```

```
}
```

1.0.22 getagrmet_lw (Source File: getdata.pl)

This subroutine makes a system call to GrADS to retrieve the AGRMET longwave radiation data.

CONTENTS:

```
sub getagrmet_lw
{
my $gradsstdout;

$gradsstdout =
`$GrADS -bl <<EOB
sdfopen http://$server:9095/dods/LIS_Forcing/AGRMET-CloudAMT/lw
set fwex
set t $index
set x 1 1440
set y 1 600
set fwrite -be $fullfilename
set gxout fwrite
d lwl
d lwm
d lwh
quit
EOB`;
open(LOG, ">>$log_dir/agrmet_lw.$iam.log");
print LOG $gradsstdout;
close(LOG);
}
```